Partitionnement de maillages sous contrainte mémoire à l'aide de la programmation linéaire en nombres entiers

Eric Angel, Cédric Chevalier, Franck Ledoux, Sébastien Morais, Damien Regnault

Compas'2016 : Parallélisme / Architecture / Système
Lorient, France, du 5 au 8 juillet 2016

Partitionnement de maillages sous contrainte mémoire à l'aide de la programmation linéaire en nombres entiers

Eric Angel[1], Cédric Chevalier[2], Franck Ledoux[2], Sébastien Morais[1,2] et Damien Regnault[1]

[1] IBISC, Université d'Evry, Evry, France
{Eric.Angel,Sebastien.Morais,Damien.Regnault}@ibisc.univ-evry.fr
[2] CEA, DAM, DIF, F-91297 Arpajon, France
{Cedric.Chevalier,Franck.Ledoux,Sebastien.Morais}@cea.fr

Résumé
L'exécution de simulations numériques sur des supercalculateurs nécessite de distribuer les données à traiter sur les différentes unités de calcul disponibles. Cette étape est réalisée à l'aide d'outils de partitionnement classiques, qui ont pour but d'équilibrer les temps de calcul sur chaque unité. Cependant, l'espace mémoire de chaque unité de calcul ayant tendance à diminuer sur les calculateurs modernes, il devient nécessaire de prendre en compte la consommation mémoire de manière explicite lors du calcul du partitionnement. Or les approches classiques ne le permettent pas car une grande partie des simulations mettent en œuvre des schémas numériques de type éléments ou volumes finis. Ces méthodes utilisent une distribution des données avec recouvrement (mailles fantômes) et cette duplication de données n'est pas représentable par un simple partitionnement de graphes. Nous proposons une nouvelle modélisation du problème de partitionnement de maillages avec une prise en compte explicite de la contrainte mémoire et, à l'aide de la programmation linéaire en nombres entiers, nous la comparons avec les formulations classiques du partitionnement.

Mots-clés : Partitionnement, maillage, programmation linéaire, CPLEX.

1. Introduction

Réaliser une simulation numérique sur un supercalculateur nécessite de nombreux composants logiciels dont font partie les outils de partitionnement [4, 6]. L'objectif de ces derniers est de répartir les données et les traitements de la simulation sur plusieurs unités de calcul (UC), et ce, en tenant compte de paramètres tels que l'équilibrage de la charge, l'ordonnancement des tâches et l'occupation mémoire. Or ce dernier critère est peu ou pas du tout pris en compte par les approches actuelles, ce qui peut poser d'importants problèmes sur des clusters où la mémoire par cœur tend à diminuer de plus en plus. En outre, une caractéristique intrinsèque au schéma numérique implanté n'est pas considérée : pour une méthode de type éléments ou volumes finis, les calculs effectués au sein d'une maille nécessitent d'accéder aux données portées par des mailles voisines [1]. Considérons l'exemple d'une simulation basée sur un tel schéma numérique, le domaine d'étude y est alors discrétisé à l'aide d'un maillage et c'est ce dernier qui est physiquement distribué entre les UC (voir Fig. 1 pour une illustration en dimension 2).

1. Le voisinage nécessaire correspond au *stencil* de la méthode.

En pratique, les calculs sont effectués au sein de chaque maille et selon le *stencil* du schéma numérique employé, le calcul effectué en une maille requiert des informations portées par un voisinage plus ou moins grand de mailles. L'approche standard est alors de disposer de ces voisinages localement à l'UC (voir Fig. 1-b).

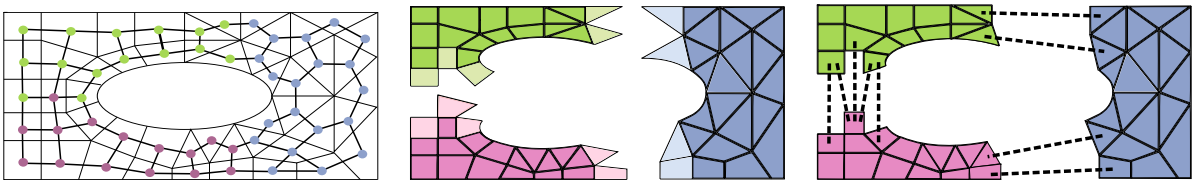


FIGURE 1 – *En (a), le maillage initial et le graphe dual associé pour le partitionnement sur 3 UC ; En (b), le partitionnement obtenu avec mailles fantômes localement à chaque UC pour un voisinage par arête ; En (c), le partitionnement sans mailles fantômes tenant compte uniquement des coûts de communication entre UC.*

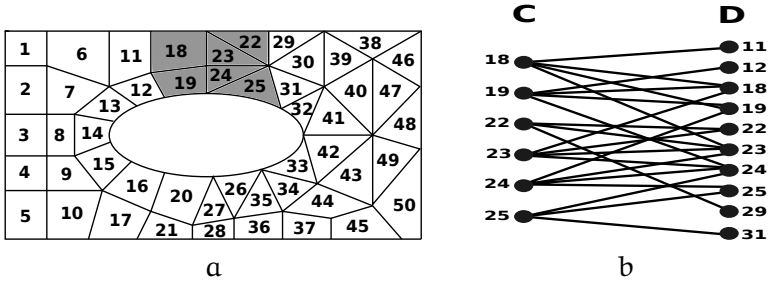
La distribution du domaine de calcul sur les k UC est réalisée par des méthodes classiques de partitionnement qui minimisent les coûts de communication entre parties (voir Fig. 1-c où les communications sont illustrées). Cependant, il n'y a pas de lien direct entre le volume de communication et la consommation mémoire sur chacune des unités de calcul. Pour considérer l'occupation mémoire des mailles où les calculs sont effectués et leurs voisines, il faut être capable de modéliser le partitionnement des calculs tout en distribuant les données du maillage avec recouvrement. C'est ce problème que nous abordons ici en tenant compte de manière explicite de l'occupation mémoire parmi les contraintes à respecter pour partitionner le maillage.

Pour cela, nous introduisons le problème de partitionnement de maillage sous contraintes mémoire qui est présenté à la section 2 et nous le modélisons sous la forme d'un programme linéaire en nombres entiers à la section 3. Enfin, à la section 4, nous comparons des résulats exacts obtenus avec l'outil CPLEX [2] pour notre nouveau modèle de partionnement et des approches classiques telles que le partitionnement de graphes [5] et d'hypergraphes [1].

2. Formulation du problème de partitionnement de maillage sous contraintes mémoire

Nous avons introduit les notions de partitionnement et de distribution, que nous précisons ici. Pour un ensemble quelconque P, soit la famille $\pi = (P_1, \ldots, P_k)$ de sous-ensembles de P. Cette famille π est appelée *partition* de P si et seulement si $P = \cup_{i=1}^{k} P_i$ et $P_i \cap P_j \neq \emptyset, \forall (P_i, P_j) \in \pi^2$ avec $i \neq j$. Les sous-ensembles P_i sont alors appelés les *parties* de la partition π. Une famille π est appelée *distribution* de P si elle vérifie seulement $P = \cup_{i=1}^{k} P_i$. Certains parlent dans ce dernier cas de partition avec recouvrement, un élement $p \in P$ pouvant appartenir à plusieurs parties.
Dans le cas du partitionnement de maillage, les approches classiques reposent sur un partitionnement du graphe dual qui produit une partition des cellules du maillage. La représentation de cette partition en mémoire n'est cependant qu'une distribution, à cause de la duplication des mailles fantômes.

Dans le cas du partitionnement de maillage sous contraintes mémoire, nous distinguons, dès la formulation, le partitionnement des calculs de la distribution des données.



a b

FIGURE 2 – *Graphe biparti* $B = (C, D, E)$ *associé au maillage. Pour des raisons de lisibilité, uniquement la partie grisée du maillage (*a*) est représentée dans le graphe (*b*). Ici, nous considérons que le calcul d'une maille* i *nécessite la connaissance des informations mémoires de* i *et des mailles partageant une arête avec* i*. Les calculs sont ainsi modélisés par les sommets de* C*, où* $c_i \in C$ *représente les calculs numériques associés à la maille* i*, les informations dont nous avons besoin pour le calcul de l'ensemble des mailles grisées sont modélisées par les sommets de* D*, où* $d_i \in D$ *modélise les données de la maille* i*. Les arêtes entre les sommets de* C *et ceux de* D *correspondent au voisinage par arête dans le maillage.*

Contrairement aux approches suivies habituellement pour le partitionnement de graphes [5] et d'hypergraphes [1] qui utilisent un graphe (ou hypergraphe) dual associé au maillage, nous modélisons notre problème en représentant le maillage à l'aide d'un graphe biparti $B = (C, D, E)$ où l'ensemble des sommets C, appelés *sommets calculs*, modélise les mailles du maillage dont on désire réaliser les calculs ; l'ensemble des sommets D, appelés *sommets données*, modélise les données des mailles du maillage nécessaires pour la réalisation des calculs ; et l'ensemble des arêtes E modélise les besoins d'information des sommets de C vis-à-vis des sommets de D, c'est-à-dire les dépendances entre C et D. Un exemple illustrant cette représentation est fourni à la figure 2 où nous considérons uniquement une partie du maillage M_Ω. Pour la suite, nous notons $\mathcal{N}(c) \subseteq D$ l'ensemble des sommets données qui appartiennent au voisinage du sommet calcul c dans $B = (C, D, E)$.

Formellement, si on note $\omega_c : c \in C \to \mathbb{R}^+$ la fonction qui associe un coût à chaque sommet calcul, $\omega_d : d \in D \to \mathbb{R}^+$ la fonction qui associe un poids à chaque sommet données, et M l'ensemble des UC ayant une capacité mémoire $W_k \in \mathbb{N}$, le problème de partitionnement de maillage sous contraintes mémoire correspond à la recherche de $\pi = (C_k)_{1 \leq k \leq |M|}$ la famille de sous-ensemble de C telle que :

1. la charge de calcul de l'UC la plus chargée, notée C_{max}, soit minimisée :

$$\min_{\pi=(C_k)\in\Pi} \max_{k\in M} \sum_{c\in C_k} \omega_c(c);$$

2. les données nécessaires à chaque UC puissent être stockées en mémoire :

$$\sum_{d\in D_k} \omega_d(d) \leq W_k, \ \forall k \in M,$$

où $D_k = \bigcup_{c\in C_k} \mathcal{N}(c), \ \forall k \in M$, est l'ensemble des données nécessaires à l'UC k pour réaliser les calculs qui lui sont attribués.

3. Modélisation du partitionnement de maillage sous contraintes mémoire à l'aide de la programmation linéaire en nombres entiers

L'utilisation de la programmation linéaire en nombres entiers (PLNE) [3] fournit un formalisme permettant une résolution exacte du problème de minimisation étudié. Pour cela, nous introduisons les inconnues $x_{c,k}$ et $y_{d,k}$ telles que : $x_{c,k}$ vaut 1 si le sommet « calcul » $c \in C$ est effectué par l'UC $k \in M$, et 0 sinon ; $y_{d,k}$ vaut 1 si le sommet « donnée » $d \in D$ est stocké en mémoire par l'UC $k \in M$ et 0 sinon. Notre problème peut alors se modéliser à l'aide du programme linéaire suivant :

`Fonction :`	`Makespan`	$\min \mathbf{C_{max}}$		
`Contraintes :`	`Affectation`	$\sum_{k \in M} x_{c,k} = 1$	$\forall c \in C$	**(1)**
	`Coût`	$\sum_{c \in C} x_{c,k}\, \omega_c(c) \leq \mathbf{C_{max}}$	$\forall k \in M$	**(2)**
	`Mémoire`	$\sum_{d \in D} y_{d,k}\, \omega_d(d) \leq W_k$	$\forall k \in M$	**(3)**
	`Fantômes`	$x_{c,k} \leq y_{d,k}$	$\forall c \in C, \forall d \in \mathcal{N}(c), \forall k \in M$	**(4)**
	`Intégrité`	$x_{c,k} \in \{0;1\}$	$\forall c \in C, \forall k \in M$	**(5)**
	`Intégrité`	$y_{d,k} \in \{0;1\}$	$\forall d \in D, \forall k \in M$	**(6)**

La première ligne est la fonction objectif du programme linéaire en nombres entiers, à savoir que l'on cherche une distribution des sommets calculs, minimisant la charge de calcul de l'UC la plus chargée. Nous avons ensuite différentes contraintes traitant de la distribution et de la mémoire. La contrainte (1) impose que chaque sommet calcul soit affecté à une UC. De plus, puisque la contrainte (5) impose que $x_{c,k} \in \{0;1\}\ \forall c \in C$ et $\forall k \in M$, l'affectation se fait sur une et une seule UC. La contrainte (2) impose que le coût des calculs associés à chaque UC soit majoré par C_{max}. La contrainte (3) impose que l'occupation mémoire des données nécessaires à chaque UC k ne dépasse pas sa capacité mémoire W_k. La contrainte (4) impose que si une UC se voit affecter le sommet calcul c alors elle doit disposer localement des données nécessaires aux calculs de c, contenues par les sommets données voisins de c. Les contraintes (5) et (6) imposent que les variables x et y ont pour valeur 0 ou 1. Une solution de ce programme linéaire optimise le coût calculatoire par le biais des variables $x_{c,k}$, en assurant que la taille mémoire incluant les mailles fantômes n'excède pas la capacité mémoire de chaque UC, grâce aux variables auxiliaires $y_{d,k}$.

4. Étude comparative en Programmation Linéaire en Nombres Entiers

Afin d'illustrer les bénéfices de notre modéle, nous avons étudié les différences entre des solutions optimales obtenues pour des problèmes de partitionnement classiques et le nôtre. Pour cela, nous avons modélisé sous la forme d'un PLNE les problèmes usuels [2] que sont le partitionnement de graphes [5] et le partitionnement d'hypergraphes [1]. Dans cette section, nous comparons ces différents modèles sur deux cas tests 2D : le cas (1) est un maillage quadrangulaire de 312 mailles où nous considérons un voisinage par arête à distance 2 [3] ; le cas (2) est

2. Faute de place, ces modèles et leurs PLNE respectifs ne sont pas présentés dans ce papier.
3. C'est-à-dire un voisinage où deux mailles sont voisines si elle partagent une arête commune ou partagent une arête avec la même maille.

un maillage triangulaire de 310 mailles avec un voisinage par arête à distance 1. Dans les deux cas, les répartitions des poids mémoires et des coûts calculatoires ne sont pas homogènes (voir Fig. 3 pour le cas (1)). Ces répartitions très hétérogènes proviennent de la nature des simulations de physique particulaire. Pour obtenir des solutions optimales, nous avons utilisé l'outil

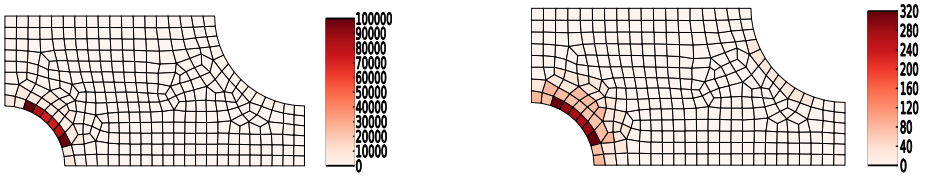


FIGURE 3 – *Distribution des coûts de calcul en* (a) *et des poids mémoire en* (b) *du cas* (1). *De façon usuelle, il existe une corrélation entre la quantité de calcul et la quantité de données associées à une maille.*

CPLEX [2] pour résoudre de manière exacte nos différents problèmes. Le nombre d'UC est fixé à 3 et pour les approches classiques, la tolérance pour le déséquilibre maximal [4] est 1.05. Préalablement à nos comparaisons, nous avons déterminé, grâce à l'outil CPLEX, la capacité mémoire minimale M_{min}, homogène à chaque UC et nécessaire à l'existence d'une solution. Ensuite, afin de disposer d'un espace de solutions assez large, nous considérons qu'une solution est valide vis-à-vis de la capacité mémoire des UC si elle ne dépasse pas une occupation mémoire de $1.2 \times M_{min}$ par UC. Les capacités mémoires des UC sont ainsi respectivement fixées à 3228 et 2566 pour les cas (1) et (2).

Pour les deux cas considérés, la figure 4 regroupe les résultats obtenus pour chaque problème, où nous évaluons aussi les charges calculatoires et mémoires de chaque UC, et illustrons la capacité des UC par une ligne pointillée.

Le premier constat est que les approches classiques ne permettent pas de respecter la contrainte mémoire. Pour le cas (2), même en ignorant l'impact des mailles fantômes, cette contrainte n'est pas respectée ! Au contraire, notre approche respecte bien le critère mémoire pour les deux cas tests, en prenant en compte la duplication des mailles fantômes.

Le deuxième constat est que le coût de calcul est lui assez similaire entre les méthodes, en particulier il n'y a pas de dégradation de notre méthode par rapport aux approches classiques. Il est donc possible de respecter les contraintes mémoire sans nuire au coût global de la simulation.

En pratique, on observe que le surcoût des mailles fantômes peut être vraiment trés important dans le cas des approches classiques ce qui peut être particulièrement préjudiciable pour l'exécution de simulation si la politique d'attribution des ressources est très stricte sur les questions mémoire. Notre approche permet de répondre à cette contrainte, sans pénaliser l'utilisateur sur le temps de restitution de la simulation.

5. Conclusion et perspectives

Dans ce papier, nous avons décrit et formalisé sous forme d'un PLNE un nouveau problème de partitionnement de maillages. Comparé aux approches usuelles, celui-ci tient compte de

4. Ce paramètre de la contrainte d'équilibre de charge permet de garantir que les parties aient sensiblement le même poids.

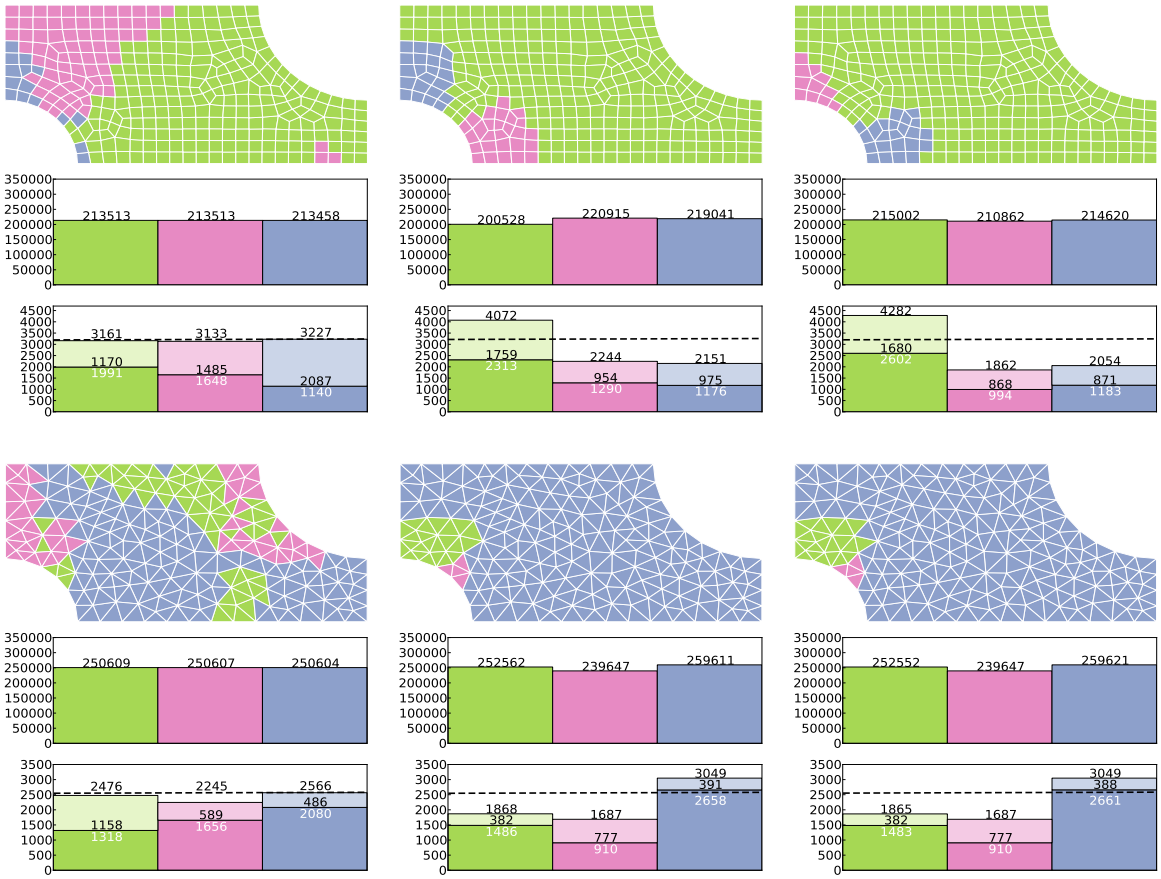


FIGURE 4 – *Résultats obtenus pour les cas* (1) *et* (2) *pour notre problème (à gauche), pour le partionnemnet de graphes (milieu) et d'hypergraphes (à droite). A chaque fois, sont founis une vue du maillage partitionné (sans les fantômes), le temps de calcul sur chaque UC et l'occupation mémoire sur chaque UC (avec en couleur foncée les mailles locales, en couleur claire les mailles fantômes et en ligne pointillée la capacité mémoire des UC).*

contraintes mémoire induites par l'architecture d'exécution (mémoire disponible sur chaque UC) et par le schéma numérique implanté (forme des stencils et mailles fantômes). Les comparaisons expérimentales menées avec l'outil CPLEX sur des maillages de petite taille montrent le gain potentiel de ce modèle. Ce gain sera d'autant plus grand pour des maillages 3D de grande taille où la proportion de mailles fantômes est bien plus élevée. L'une des perspectives à ce travail est l'étude d'heuristiques permettant la résolution de ce problème de partitionnement sous contraintes mémoire. Il serait tout particulièrement intéressant d'obtenir des algorithmes retournant des solutions qui respectent les contraintes mémoire de notre problème tout en étant à un facteur fixé du coût global de simulation optimal.

Bibliographie

1. Çatalyürek (Ü. V.) et Aykanat (C.). – Hypergraph model for mapping repeated sparse matrix-vector product computations onto multicomputers. *Proc. International Conference on High Performance Computing*, 1995.
2. CPLEX (I. I.). – V12. 1 : User's manual for cplex. *International Business Machines Corporation*, vol. 46, n53, 2009, p. 157.
3. Dantzig (G.). – *Linear Programming and Extensions*, 1963.
4. Karypis (G.) et Kumar (V.). – A fast and highly quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing, Vol. 20, No. 1, pp. 359–392*, 1999.
5. Kernighan (B. W.) et Lin (S.). – An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 1970.
6. Pellegrini (F.) et Roman (J.). – Scotch : A software package for static mapping by dual recursive bipartitioning of process and architecture graphs. *Proceedings of HPCN'96, Brussels, Belgium. LNCS 1067, pages 493-498*, 1996.